Matrik : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer
Vol. 20, No. 1, November 2020, pp. 149~158
ISSN: 2476-9843, accredited by Kemenristekdikti, Decree No: 3/E/KPT/2019
DOI: 10.30812/matrik.v20i1.853

Integrasi Pendekatan Metode *Custom Hashing* dan *Data Partitioning* untuk Mempercepat Proses Pencarian Frekuensi *Item-set* pada Algoritma Apriori

Moch. Syahrir, Fatimatuzzahra
Universitas Bumigora, Indonesia

Article info

Article history:

Received, 8 August 2020
Revised, 5 September 2020
Accepted, 15 September 2020

Kata kunci:

Apriori
Aturan Asosiasi
FP-Growth
Hashing
Data Partisi

ABSTRAK

Data mining dengan peran asosiasi sudah banyak digunakan oleh dunia usaha, salah satu algoritma yang sering digunakan untuk aturan asosiasi adalah apriori. Namun apriori memiliki kelemahan dalam hal performa, karena pada setiap penentuan *frequent k-itemset* harus melakukan *scan database*. Hal ini akan menjadi masalah apabila kandidat *k-itemset* memiliki dimensi yang banyak. proses *scan database* yang besar akan memakan waktu yang lama dan berpengaruh pada penggunaan memori dan prosesor. Apriori sudah sering dikembangkan, salah satu yang populer adalah *Frequent Pattern* (*fp-growth)*, apriori dan *fp-growth* sama-sama merupakan algoritma untuk aturan asosiasi, hanya saja *fp-growth* menggunakan pendekatan yang berbeda dengan apriori yakni menggunakan pendekatan *Frequent Pattern Tree* (*fp-tree)*. Meski *fp-growth* memiiki performa yang bagus ketika *scan database* namun *rules* yang di hasilkan oleh *fp-growth* tidak sebaik yang di hasilkan oleh apriori. Alternatif lain yang bisa digunakan adalah metode *hashing*, hal ini bisa menjadi solusi untuk mengatasi masalah dalam proses pencarian dan penentuan *frequent k-itemset*, sehingga proses *scan database* bisa lebih cepat. Tujuan penelitian adalah memperbaiki kinerja apriori dalam proses pencarian frekuensi itemset sehingga waktu *scan database* bisa lebih cepat

Keywords:

Apriori
Association Rule
FP-Growth
Hashing
Data Partitioning

ABSTRACT

Data mining with the role of association has been widely used by the business world. One of the algorithms often used for association rules is a priori. However, a priori has a weakness in terms of performance, because every frequent k-itemset determination has to scan the database. This will be a problem if the k-itemset candidate has many dimensions. scanning large databases will take a long time and will affect memory and processor usage. Apriori has often been developed, one of the popular ones is that fp-growth, a priori and fp-growth are both algorithms for association rules, it's just that fp-growth uses a different approach from a priori, namely using the fp-tree approach. Even though fp-growth has good performance when scanning the database, the rules generated by fp-growth are not as good as those generated by a priori. Another alternative that can be used is the hashing method, this can be a solution to solving problems in the search process and determining frequent k-itemset, so that the database scan process can be faster. The research objective was to improve a priori performance in the itemet frequency search process so that the database scan time could be faster

Penulis Korespondensi:

Moch. Syahrir,
Program Studi Ilmu Komputer,
Universitas Bumigora.
Email: muhammadsyahriralfath@gmail.com

1. PENDAHULUAN

Algoritma untuk *association rules* sangat banyak diantaranya apriori, *eclat, fp-growth*, mafia dan lain-lain. Akan tetapi algoritma yang paling umum di kembangkan sebagai dasar pengembangan dan *baseline method* adalah apriori [1], sebagaimana yang telah di utarakan oleh para peneliti bahwa masalah yang paling umum pada apriori adalah *scan database* secara berulang-ulang untuk mendapatkan frekuensi *itemset* dengan kombinasi-kombinasi item tertentu, sehingga akan banyak memakan waktu dan memori komputer yang sangat besar untuk *scan database* [2]. Seiring berjalannya waktu dan berkembangnya teknologi mulai bermunculan algoritma-algoritma hasil pengembangan dari apriori tersebut dan yang paling populer adalah *fp-growth* dengan pendekatan *fp-tree*. Dengan konsep *fp-tree*, item-item sebagai fitur di bentuk dalam sebuah kerangka pohon dengan lintasan-lintasan tertentu yang sangat padat [3], sehingga *scan database* hanya dilakukan satu atau dua kali. Menurut para ahli dan peneliti *fp-growth* bukan tanpa masalah, permasalahan yang terdapat pada *fp-growth* adalah *rules* yang dihasilkan oleh *fp-growth* tidak sebaik *rules* yang dihasilkan oleh apriori [2]. Dan juga cara implementasi *fp-growth* tidak semudah cara mengimplementasikan apriori walaupun ini hal yang relative [4].

Fp-growth dikembangkan dari algoritma apriori, tentu keduanya saling melengkapi, *Fp-growth* dalam proses pencarian frekuensi *itemset* sudah sangat baik dari algoritma apriori namun *rules* yang dihasilkan tidak sebaik algoritma apriori. Permasalahan yang muncul adalah bagaimana bisa mempertahankan *rules* optimal yang di hasilkan oleh algoritma apriori dan mengurangi waktus *scan* database berulang-ulang. Salah satu alternatif dengan memanfaatkan tabel *hashing* yang di bentuk dengan menggunakan metode *hashing* dengan *key*, dihasilkan dari sebuah perhitungan fungsi yang bervariasi. Kita bisa menentukan *key* dalam tabel *hashing* lalu memanggil *key* tersebut untuk melakukan pencarian dan penghapusan data [5] pemanfaatan metode *hasing* untuk menyimpan alamat *key* dalam menentukan frekuensi *itemset*.

Metode *hashing* adalah teknik untuk melakukan penambahan data, penghapusan, dan pencaraian data dengan teknik menelusuri alamat *key* yang didapat dari sebuah aturan fungsi [5]. Fungsi *hashing* cukup banyak, kita tidak perlu menggunakan semua fungsi tersebut, disesuaikan dengan objek penelitian sebagai salah satu yang bisa di gunakan adalah *Modular Aritmatic* yakni melakukan konversi data ke bilangan bulat lalu di bagi dengan ukuran tabel *hash* dan mengambil hasil sisa bagi sebagai *indeks* atau *key* [6]. Sebuah fungsi *hashing* yang bagus memiliki dua kriteria yakni harus cepat dapat dihitung dan harus meminimalkan terjadinya *collision*. Dalam beberapa jurnal penelitian fungsi *hashing* sudah banyak di modifikasi, tujuannya tidak lain yakni meminimalkan terjadinya *collision*dan mudah dapat di hitung.

Data *mining* adalah disiplin ilmu yang mempelajari metode untuk mengekstrak pengetahuan atau menemukan pola dari suatu data yang besar [7]. Tahap-tahap data *mining*:

1. Seleksi Data

   Data tidak semuanya dipakai, oleh karena itu hanya data yang sesuai untuk dianalisis yang akan diambil dari *database.*

2. *Preprosesing* Data

   Data yang sudah di pilih di lakukan pembersihan data, dan integrasi data untuk menggabungkan beberapa data ke dalam *database* baru.

3. Transformasi Data

   Data diubah dan digabungkan ke dalam format yang sesuai untuk proses dalam data *mining*. Transformasi dapat di lakukan dengan cara sebagai berikut:

1. *Smooting* untuk menghilangkan *noise* dari data
2. *Attribute Construction*, dimana attribut baru di buat atau ditambahkan untuk membantu proses *mining*
3. *Aggreation*, dimana ringkasan atau operasi agrerasi diterapkan pada data.
4. *Normalization*, dimana data attribut dibuat dalam skala tertentu sehingga menjadi kisaran data yang lebih kecil dan sebaran datanya tidak terlalu jauh.
5. Proses *Mining m*erupakan suatu proses utama saat metode diterapkan untuk menemukan pengetahuan berharga dan tersembunyi dari data. Terdapat berbagai macam teknik dalam data *mining*, dimana pemilihannya bergantung pada tujuan dan proses pencarian pengetahuannya.
6. Evaluasi Pola (*Pattern Evaluation*) Untuk mengidentifikasi pola-pola menarik ke dalam *knowledge based* yang ditemukan. Kemudian dilakukan visualisasi dan penyajian pengetahuan mengenai metode yang digunakan untuk memperoleh pengetahuan yang berguna bagi pengguna.

Algoritma apriori adalah salah satu algoritma *association rules* dengan teknik pengambilan data menggunakan pendekatan aturan asosiatif untuk menentukan hubungan asosiasi suatu kombinasi itemset. Penting tidaknya suatu aturan asosiatif dapat diketahui dengan dua parameter yaitu *support* dan *confidence* [7]. *Support* (nilai penunjang) adalah persentase kombinasi item tersebut dalam *database*. *Confidence* (nilai kepastian) adalah kuatnya hubungan antar-item dalam aturan asosiasi. Sebuah aturan asosiasi dikatakan *interesting* jika nilai *support* adalah lebih besar dari minimum *support* dan juga nilai *confidence* adalah lebih besar dari minimum *confidence*. Sementara untuk menguji nilai kevalidan hubungan antar item menggunakan *lift rasio*. Adapun formula yang digunakan untuk meghitung nilai *support*, *confidence*, dan *lift* rasio pada persamaan (1), (2), dan (3).

1. Formula *Support*:

$$Support(A,B) = \frac{Jumlah\ Transaksi\ A\ Dan\ B}{\sum Transaksi} x\ 100\% \quad (1)$$

2. Formula *Confidence*:

$$Confidence(A,B) = \frac{Jumlah\ Transaksi\ A\ Dan\ B}{\sum TransaksiA} x\ 100\% \quad (2)$$

3. Formula *Lift Rasio*:

$$Lift\ rasio = \frac{Support(A,B)}{Support\ A\ x\ Support\ B} \quad (3)$$

Algoritma *Fp-growth* adalah salah satu algoritma dari teknik *association rules* yang dapat digunakan untuk menentukan himpunan data yang paling sering muncul *(frequent itemset)* dalam sebuah kumpulan data dengan pendekatan pada konsep *fp-tree* [8].

Cara kerja *Fp-Tree*:

1. *Dataset*

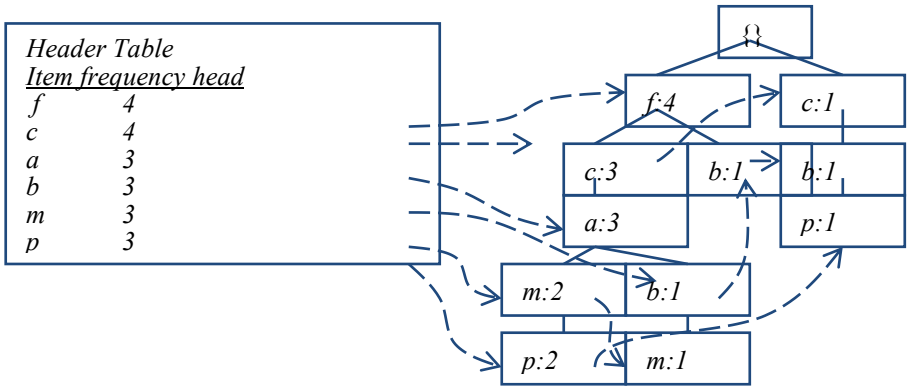
Tabel 1. Kontruksi *Fp-Tree* Dari Transaksi *Database*

TID	*Item Bougth*	*Frequent Items*
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, i, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

2. Menentukan minimum *support*

   Minimum *support* = 3

3. *Scan database* untuk mendapatkan frekuensi 1 *itemset*
4. Pengurutan nilai frekuensi dari nilai paling tinggi sampai nilai yang paling rendah
5. *Scan database* lagi untuk membentuk *fp-tree*



Gambar 1. Pembentukan *Fp-Tree*

Metode *hashing* adalah salah satu metode yang bisa digunakan untuk membentuk tabel *array* dan membentuk alamat *key* dengan fungsi *hashing* [5].

1. *Hashing* digunakan untuk menyimpan data yang cukup besar pada ADT (*Abstract Data Type*) yang di sebut tabel *Hash*.
2. Ukuran tabel *Hash* biasanya lebih besar dari jumlah data yang akan di simpan.
3. Fungsi *Hashing* memetakan elemen pada *indeks* dari tabel *Hash*.
4. *Hash Table* adalah *Array* dengan sel-sel yang ukuranya telah di tentukan dan dapat berisi data atau *key* yang memiliki kesamaan dengan data. Namun kita bisa juga mengganti tabel *array* dalam bentuk tabel sementara di *mysql* yang memiliki *record–rocord* dan *primary key* yang sesuai dengan data tersebut sebagai acuan untuk *insert, update, delete, search,* dan *link.*
5. Untuk menambahkan data atau pencarian ditentukan *key* dari data tersebut dan digunakan sebuah fungsi *hashing* untuk menetapkan lokasi *key* tersebut.
6. Untuk memetakan setiap *key*, fungsi *hashing* dapat digunakan pada bilangan dalam rentang 0 hingga *Hash-Size* -1.

Fungsi *hashing* cukup banyak, tidak perlu menggunakan semua fungsi tersebut, serta disesuaikan dengan objek penelitian. Salah satu fungsi *hasing* yang bisa digunakan adalah *Modular Aritmatic* yaitu dengan cara melakukan konversi data ke bilangan bulat, dibagi dengan ukuran *Hash Table* dan mengambil hasil sisa bagi sebagai *indeks* / alamat *key*.

Sebuah fungsi *hashing* yang bagus memiliki dua kriteria yakni harus dapat cepat dihitung dan harus meminimalkan terjadinya *collisoin*. *Collisoin* adalah jika dua buah *key* atau lebih di petakan pada sel data yang sama. Ada dua strategi umum untuk meminimalisir terjadinya *collision* yakni *Close Hashing* (*Opening Addres*) dan *Open Hashing* (*Chaining*).

Salah satu rumus untuk pencarian fungsi *hashing* yang digunakan oleh peneliti-peneliti dalam menentukan alamat *key*sebagai pemetaan data hasil *scan database* [9]. Formula standar fungsi *hashing* ditunjukkan pada persamaan 4.

$$H_i(x) = \big(Hash(x) + f(i)\big) Mod\ H - Sizef(0) = 0 \quad (4)$$

f(i) = digunakan untuk mengatur strategi *collision resolution*.

Namun, jika di implementasikan akan terjadi beberapa *collision* sehingga dalam penelitian ini fungsi *hashing* yang digunakan sudah dimodifikasi, tujuannya untuk mempermudah perhitungan dan meminimalkan terjadinya *collision*. Secara rinci bisa di lihat pada persamaan (5).

Penentuan alamat *key* untuk *itemset*.

$$H(c) = \sum_{k=2}^{n} \big(ip * 10^{(k-1)}\big) + (ic) Mod\ 2 * m + 1 \quad (5)$$

H(c) adalah alamat *hashing, n* adalah jumlah iterasi, *k* adalah iterasi ke, *ip* adalah item *Premises, ic* adalah item *Conclusion,* dan *m* adalah jumlah transaksi.

Permasalahan umum yang terdapat dalam data *mining* adalah banyaknya atribut sehingga kita bisa mengenal seleksi fitur dan ekstraksi fitur, akan tetapi dalam asosiasi melakukan hal ini sangat beresiko sebab item-item dari *database* akan menjadi atribut *dataset* yang akan diolah dan akan dilakukan pencarian frekuensi kemunculannya dalam sebuah tabel transaksi [9], item akan mejadi atribut yang saling berkorelasi antara satu dengan yang lainnya, apabila fitur kita kurangi dan kita ekstraksi tentu *rules* yang dihasilkan dari proses tersebut tidak optimal [10]. Sehingga teknik yang bisa digunakan adalah membagi atribut-atribut tersebut dan dikelompokan untuk mempercepat proses *scan database* tanpa memangkas pembentukan *rules* serta tetap mempertahankan *support* dan *confidence* yang benar dan juga dalam penelitian ini *item partition* berguna untuk membatasi pembentukan alamat dari fungsi *hash* supaya tidak terjadi *collison*.

Untuk menentukan *state of the art* dari penelitian ini penulis melakukan review terhadap beberapa paper yang relevan dengan topik yang di bahas. Ada beberapa jurnal yang membahas tentang komparasi algoritma *association rules* dan proses peningkatan kinerja dari pada algoritma-algoritma *association rule*, dan juga membahas tentang algoritma apriori di gabungkan dengan metode *hashing* sebagai alternatif lain dalam pengembangan ataupun menncari kebaharuan algoritma apriori sehingga kinerja algoritma apriori menjadi lebih baik. Sebagai *state of the art* untuk mengembangkan hasil riset dan *paper* oleh penulis-penulis sebelumnya yang menjadi bahan pengembangan sebagai berikut:

Performance Comparison Of Hahshing Algoritm With Apriori [11].Dalam penelitian ini, rathin dan baskaran membuat *fp-tree* dari kumpulan dataset hasil *scanning* tersebut di masukan ke dalam tabel *hash* yang sebelumnya sudah di tentukan alamatnya. Sebagai *baseline method* mereka menggunakan apriori tradisional.

An effective Hash-Based Algorithm For Mining Association Rule [12].Dalam penelitian ini, phils. S. Yu membentuk tabel hash lalu memasukan data hasil scandatabase ke dalam tabel hash tersebut yang sebelumnya sudah di pangkas, dan untuk meminimalisir terjadinya collisiontabel hashmerubah fungsi dengan menambahkan konstanta ke dalam fungsi hash tersebut.

Untuk menentukan *state of the art* dari penelitian ini penulis melakukan review terhadap beberapa paper yang relevan dengan topik yang di bahas. Ada beberapa jurnal yang membahas tentang komparasi algoritma *association rules* dan proses peningkatan kinerja dari pada algoritma-algoritma *association rule* dan juga membahas tentang algoritma apriori digabungkan dengan metode *hashing* sebagai alternatif lain dalam pengembangan ataupun mencari kebaharuan algoritma apriori sehingga kinerja algoritma apriori menjadi lebih baik. Sebagai *state of the art* untuk mengembangkan hasil riset dan *paper* oleh penulis-penulis sebelumnya yang menjadi bahan pengembangan sebagai berikut:

Penelitian [11] membuat *fp-tree* dari kumpulan dataset hasil *scanning* tersebut dimasukan ke dalam tabel *hash* yang sebelumnya sudah ditentukan alamatnya. Sebagai *baseline method* mereka menggunakan apriori tradisional. Penelitian [12] membentuk tabel *hash* lalu memasukan data hasil *scan database* ke dalam tabel *hash* tersebut yang sebelumnya sudah di pangkas dan untuk meminimalisir terjadinya *collision* tabel *hash* merubah fungsi dengan menambahkan konstanta ke dalam fungsi *hash* tersebut. Dalam penelitian [13] membentuk *fp-tree* untuk membuat lintasan korelasi antar item serta di implementasikan dalam bentuk aplikasi data *mining* yang bisa menampilkan hasil secara *real time.*

Penelitian [14] menggunakan tabel *hash* untuk mempermudah penyimpanan data hasil *scan* dan meminamalkan kolom dalam tabel *hash*. Penelitian [15] lebih fokus pada penentuan fungsi dengan teknik *linear probing* dalam menentukan alamat *key* pada tabel *hash*.

Dalam penelitian ini ada dua pengembangan yang paling mendasar yaitu mencari alternatif lain untuk algoritma apriori agar memiliki kinerja yang lebih baik dengan mempertahankan *rules* yang baik dari algoritma apriori dan juga memodifikasi fungsi pada metode *hashing* agar *collision* yang terjadi dapat diminimalisir.

2. METODE PENELITIAN

2.1. Sumber Data

Data yang di peroleh adalah data *private* yang sudah di validasi oleh instansi terkait. Dalam hal ini KOPEGTEL (Koperasi Pegawai Telkom) Kota Mataram – Nusa Tenggara Barat (NTB). Total *record* data lebih kurang lebih 500 *record*, sementara item data lebih dari 50 item, data yang di ambil adalah data proses transaksi penjualan pada KOPEGTEL Kota Mataram. Untuk memaksimalkan kinerja algoritma *association rules* item akan dipartisi maksimal 15 item karena kelemahan data mining adalah ketika dimensi atau item tersebut berjumlah sangat banyak maka proses *scan* data memakan waktu yang lama. Sebelum data digunakan data dilakukan *preprosesing*, model *preprosesing* dilakukan dengan membentuk tabel *binari binominal*.

2.2. Metodologi Penelitian

Algoritma yang digunakan dalam penelitian ini adalah algoritma apriori, algoritma apriori sendiri adalah bagian dari algoritma-algoritma *association rule*. Kelemahan algoritma apriori adalah pada saat *scan database* dalam proses pencarian frekuensi itemset akan tetapi kelebihannya mampu menghasilkan *rule* yang optimal. Oleh sebab itu dalam penelitian ini penulis menggunakan metode *hashing* untuk memperbaiki kinerja algoritma apriori dalam proses pencarian frekuensi. Metode *hashing* bisa digunakan untuk menentukan alamat key, dimana alamat key ini digunakan untuk menyimpan frekuensi itemset hasil *scan* database. Adapun tahapan-tahapan dalam penelitian adalah pengumpulan data serta proses *preprosesing*, proses asosiasi dengan algoritma apriori tradisional, proses asosiasi dengan algoritma apriori +*hashing*, proses asosiasi dengan algoritma *fp-growth* dalam *tools rapidminer*, dan terakhir dibuatkan grafik perbandingan.

Tahap pertama yang dilakukan yaitu mengumpulkan data. Data yang diperoleh sebanyak 500 *record* serta lebih dari 50 item, akan tetapi maksimal item yang dipartisi adalah 15 item. Tujuannya untuk mempermudah proses–proses selanjutnya. Data yang diperoleh adalah data *private* dari KOPEGTEL (Koperasi Pegawai Telkom) Kota Mataram – Nusa Tenggara Barat (NTB) yang telah di validasi oleh pihak terkait. Sebelum masuk tahap kedua data dilakukan *preprosesing* terlebih dahulu, model *preprosesing* dilakukan dengan membentuk tabel *binari binominal*.

Tahap kedua di lakukan proses asosiasi menggunakan algoritma apriori tradisional dengan menggunakan data yang telah di *preprosessing* sebelumnya serta menentukan *treshold* frekuensi minimum *support* bertujuan

sebagai *prunning* agar hasil *scan* tidak melebar tampa ada pemangkas bagi korelasi yang di anggap di bawah *treshold* minimum *support*. Serta melakukan perhitungan *support* dan *confidence.* Untuk proses yang dilakukan beberapa tahap dengan variasi *record* dan *threshold* yang berbeda-beda. Sementara proses yang dilakukan sebanyak 4 iterasi *itemset*.

Tahap ketiga dilakukan proses asosiasi menggunakan algoritma apriori + *hashing* dengan menggunakan data dan menentukan *threshold* yang sama dengan tahapan sebelumnya. Dengan begitu kita dengan mudah dapat membandingkan hasil *support* dan *confidence* antara kedua algoritma tersebut serta menghitung perbedaan waktu yang di butuhkan dalam hal *scan database*.

Tahap ke empat dilakukan proses asosiasi menggunakan algoritma *fp-growth* yang memanfaatkan *tools rapidminner* dengan data dan *treshold* yang sama seperti tahap-tahap sebelumnya. Sehingga kita bisa membandingkan nilai *support* dan *confidence* dengan algoritma apriori + *hashisng* maupun dengan algoritma *fp-growth* yang telah tersedia pada *rapidminner*.

Setelah semua data hasil asosiasi ketiga algoritma pada tahap-tahap tersebut di buatkan dalam bentuk tabel untuk mempermudah dalam membaca dan menganalisa perbedaan.

3. HASIL DAN ANALISIS

3.1. Pengujian Program.

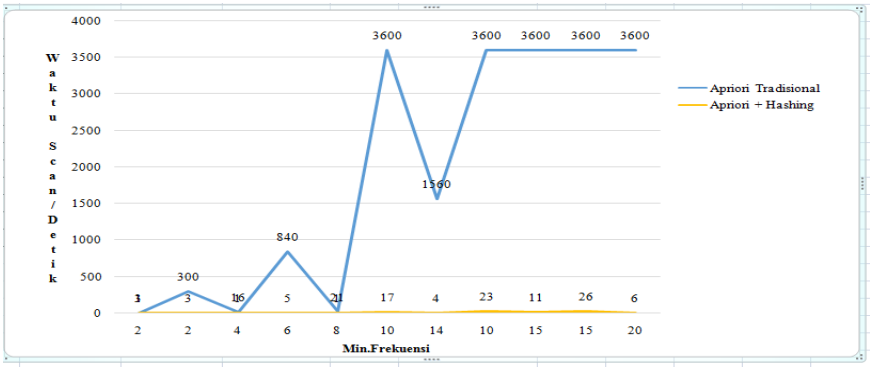
Pengujian program di lakukan dengan acuan perbandingan waktu *scan*, perbedaan nilai *support, confidence,* dan evaluasi nilai *liftrasio.* dengan ukuran data yang bervariasi baik untuk *record* maupun fitur. Sebagai bahan pengujian menggunakan sampel data *private* yang sudah di validasi oleh instansi terkait. Dalam hal ini KOPEGTEL (Koperasi Pegawai Telkom) Kota Mataram - NTB.

3.2. Pengujian Waktu *Scan Database*

Dalam tahap ini di lakukan perhitungan waktu scan database antara algoritma apriori tradisional dengan algoritma apriori + *hashing*.

1. Pengujian untuk 2-*Itemset*

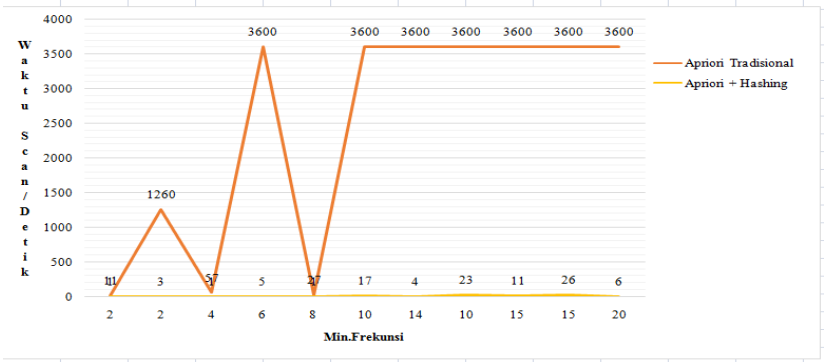
Pada Gambar 2 kita bisa melihat grafik perbedaan waktu yang sangat siginifikan dalam proses pencarian frekuensi itemset antara algoritma apriori tradisional dengan algoritma apriori + hashing. Semakin banyak minimum frekuensi sebagai *threshold*-nya maka waktu pencarian frekuensi akan semakin lama. Karena harus terus mencari sampai batas *threshold* yang di tetapkan. Selain itu kita bisa melihat meskipun minimum frekuensi lebih kecil itu terkadang juga membutuhkan waktu yang lama dalam pencarian frekuensi *itemset*, ini disebabkan oleh item yang banyak. Item dalam algoritma-algoritma *association rulesi* adalah atribut atau dimensi dalam istilah data mining.



Gambar 2. Pengujian Waktu *Scan 2-Itemset*

2. Pengujian untuk 3-*Itemset*

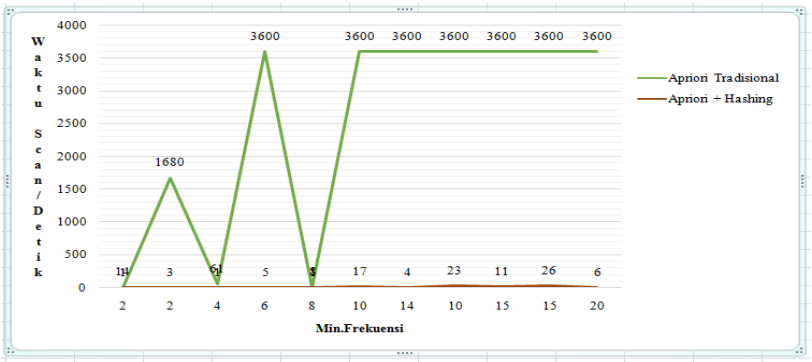
Pada Gambar 3 kita bisa melihat perbedaan waktu *scan database* yang sangat signifikan sama dengan pada Gambar 2. Akan tetapi rata-rata waktu yang dibutuhkan semakin besar karena *itemset* bertambah.



Gambar 3. Pengujian Waktu *Scan 3-Itemset*

3. Pengujian Untuk 4-*Itemset*

Dari Gambar 4 kita juga bisa melihat perbedaan waktu *scan database* yang sangat signifikan, juga karena *itemset* bertambah maka waktu *scan database* pun akan semakin lama. dengan ini kita melihat bahwa algoritma apriori yang telah di modifikasi dengan tambahan metode *hashing* mampu mempercepat proses pencarian frekuensi itemset dari *database* yang di *scan*.



Gambar 4. Pengujian Waktu *Scan 4-Itemset*

Dari Gambar 4 kita bisa melihat perbedaan waktu scan database yang sangat signifikan sama dengan Gambar 2 dan Gambar 3 akan tetapi waktu yang dibutuhkan juga semakin lama dan ini berlaku untuk kedua algoritma baik apriori tradisional maupun apriori yang di modifikasi dengan metode hashing, akan tetapi tetap apriori yang telah di modifikasi yang bisa lebih cepat dalam proses scan database. Sementara hasil support dan confidence antara apriori tradisional dengan apriori + hashing tidak ada perbedaan.

3.3. Pengujian Nilai Support Dan Confidence

Pengujian Nilai *Support* Dan *Confidence. T*ahap pengujian ini melibatkan hasil dari ke tiga algoritma asosiasi *rule* yakni algoritma apriori tradisional, apriori + *hashing*, dan *fp-growth* dengan jumlah *record* data yang sama dan juga nilai *treshold* yang sama pula. Pengujian ini menggunakan *threshold support* minimum 0.1 dan item *Premises* yakni PEPSODEN.

Tabel 2. Pengujian Nilai *Support* Dan *Confidence* 2-*Itemset* Algoritma Apriori Tradisional dan Algoritma Apriori + *Hashing*.

No	Tresh	Item 1	Item 2	*2-Itemset Apriori Tradisional*		*2 Itemset Apriori + Hashing*	
				Supp	Conf	Supp	Conf
1	0.1	Pepsoden	Deterjen	64	0.92	64	0.92
2	0.1	Pepsoden	Sabun	50	0.72	50	0.72
3	0.1	Pepsoden	Popok	47	0.68	47	0.68
4	0.1	Pepsoden	Sampo	42	0.60	42	0.60
5	0.1	Pepsoden	Celana	42	0.60	42	0.60
6	0.1	Pepsoden	Gula	Supp	Conf	Supp	Conf
7	0.1	Pepsoden	Teh	36	0.52	36	0.52
8	0.1	Pepsoden	Kopi	33	0.48	33	0.48
9	0.1	Pepsoden	Susu	33	0.48	33	0.48
10	0.1	Pepsoden	Coklat	31	0.44	31	0.44
11	0.1	Pepsoden	Gatsbi	31	0.44	31	0.44
12	0.1	Pepsoden	Aqua	31	0.44	31	0.44
13	0.1	Pepsoden	Roti	28	0.40	28	0.40
14	0.1	Pepsoden	Parfum	28	0.40	28	0.40

Tabel 3. Pengujian Nilai *Support* Dan *Confidence* 3-*Itemset* Algoritma Apriori Tradisional dan Algoritma Apriori + *Hashing*

No	Tresh	Item 1	Item 2	Item 3	3-*Itemset* Tradisional	Apriori	3-*Itemset Hashing*	Apriori +
					Supp	Conf	Supp	Conf
1	0.1	Pepsoden	Deterjen	Sabun	44	0.70	44	0.70
2	0.1	Pepsoden	Deterjen	Popok	44	0.70	44	0.70
3	0.1	Pepsoden	Deterjen	Celana	42	0.66	42	0.66
4	0.1	Pepsoden	Deterjen	Sampo	39	0.61	39	0.61
5	0.1	Pepsoden	Popok	Celana	39	0.82	39	0.82
6	0.1	Pepsoden	Gula	Deterjen	36	0.93	36	0.93
7	0.1	Pepsoden	Gula	Sabun	36	0.93	36	0.93
8	0.1	Pepsoden	Sampo	Popok	33	0.80	33	0.80
9	0.1	Pepsoden	Sabun	Pokok	31	0.61	31	0.61
10	0.1	Pepsoden	Sabun	Kopi	31	0.61	31	0.61
11	0.1	Pepsoden	Gula	Kopi	31	0.79	31	0.79
12	0.1	Pepsoden	Deterjen	The	31	0.48	31	0.48
13	0.1	Pepsoden	Deterjen	Susu	31	0.48	31	0.48
14	0.1	Pepsoden	Deterjen	Gatsbi	31	0.48	31	0.48
15	0.1	Pepsoden	Sabun	Aqua	28	0.56	28	0.56
16	0.1	Pepsoden	Gula	Coklat	28	0.71	28	0.71
17	0.1	Pepsoden	Gula	Aqua	28	0.71	28	0.71
18	0.1	Pepsoden	Sampo	Celana	28	0.67	28	0.67
19	0.1	Pepsoden	Deterjen	Kopi	28	0.43	28	0.43
20	0.1	Pepsoden	Teh	Susu	28	0.77	28	0.77
21	0.1	Pepsoden	Deterjen	Roti	28	0.43	28	0.43
22	0.1	Pepsoden	Deterjen	Coklat	28	0.43	28	0.43
23	0.1	Pepsoden	Deterjen	Aqua	28	0.43	28	0.43

Tabel 4. Pengujian Nilai *Support* Dan *Confidence* 2-*Itemset* Algoritma *Fp-Growth* Dan Algoritma Apriori + *Hashing*

No	Tresh	Item 1	Item 2	*2-Itemset Fp-Growth Rapidminer*		2 *Itemset* Apriori + *Hashing*	
					Conf	Supp	Conf
1	0.1	Pepsoden	Deterjen	64	0.92	64	
2	0.1	Pepsoden	Sabun	50	0.72	50	0.72
3	0.1	Pepsoden	Popok	47	0.68	47	0.68
4	0.1	Pepsoden	Sampo	42	0.60	42	0.60
5	0.1	Pepsoden	Celana	42	0.60	42	0.60
6	0.1	Pepsoden	Gula	39	0.56	39	0.56
7	0.1	Pepsoden	Teh	36	0.52	36	0.52
8	0.1	Pepsoden	Kopi	33	0.48	33	0.48
9	0.1	Pepsoden	Susu	33	0.48	33	0.48
10	0.1	Pepsoden	Coklat	31	0.44	31	0.44
11	0.1	Pepsoden	Gatsbi	31	0.44	31	0.44
12	0.1	Pepsoden	Aqua	31	0.44	31	0.44
13	0.1	Pepsoden	Roti	28	0.40	28	0.40

Pada tabel 2 dan table 3 kita bisa melihat bagaimana algoritma apriori + *hashing* tetap mempertahankan *support, confidence* dan *rules* yang memang di hasilkan oleh apriori tradisional yang terkenal baik akurasinya. Tetapi pada tabel 4 ada satu rule yang tidak terbentuk, nilai *support* dan *confidence* tidak masuk pada ambang *treshold* dan juga nilai support dan confidence yang di hasilkan oleh algoritma *fp-growth* dalam *tools rapidminner* sedikit berbeda dan lebih kecil, karena pada dasarnya algoritma yang digunakan dalam *tools* rapidminner tersebut adalah algoritma *fp-growth* dengan pendekatan *fp-tree* yang mana *rules* yang di hasilkan tidak sebaik algoritma apriori [11].

4. KESIMPULAN

Dari hasil yang telah di peroleh nilai *support* dan *confidence* antara algoritma apriori tradisional dengan apriori + *hashing* tidak ada perbedaan akan tetapi untuk mendapatkan hasil tersebut ke dua algoritma ini membutuhkan waktu yang berbeda dalam melakukan proses *scan* database dimana algoritma apriori + *hashing* jauh lebih baik dibanding algoritma apriori tradisional dengan perbedaan efisiensi waktu yang cukup jauh. Dan juga menjadi catatan penting semakin banyak fitur yang terseleksi akan semakin banyak waktu yang di perlukan untuk proses *scan* database.

Sementara hasil lain yang kita peroleh yakni nilai *support* dan *confidence* antara algoritma apriori + *hashing* dengan *fp-growth* terjadi perbedaan nilai walaupun pada umumnya kebanyakan yang sama. Kenapa hal ini bisa terjadi, karena konsep kerja dan pendekatan dari ke dua algoritma ini berbeda meskipun tujuannya sama yakni aturan asosiasi. *Fp-growth* melakukan pendekat *fp-tree,* dimana *fp-tree* membentuk lintasan-lintasan yang padat sehingga hasil *rules* yang di hasilkan *fp-growth* tidak sebaik apriori. Tujuan dari penelitian

ini yakni meningkatkan kinerja algoritma apriori dalam hal kecepatan proses *scan database* pada saat pencarian frekuensi itemset dengan mempertahankan hasil *rules* yang optimal dari algoritma apriori tradisional tercapai.

Untuk pengembangan di harapkan bisa menentukan banyak fitur yang paling ideal untuk proses *scan* itemset sehingga bisa lebih mengoptimalkan hasil kerja dari algoritma apriori +*hashing* ini. Dan juga di harapkan algoritma apriori +*hashing* ini di kembangkan lagi dengan teknik yang berbeda yakni pendekatan algoritma eclat +*query join*serta melakukan komprasi untuk kedua algoritma tersebut.

UCAPAN TERIMA KASIH

Penulis ucapkan terima kasih

REFERENSI

[1] X. Wu *et al.*, "Top 10 Algorithms in Data Mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[2] R. M. Chezian and K. S. Kumar, "A Survey on Association Rule Mining using Apriori Algorithm," *International Journal of Computer Applications*, vol. 45, no. 5, 2012.

[3] M. North, *Data Mining for the Messes*. Utah: CreateSpace Independent Publishing Platform, 2016.

[4] L. Jian-ping, W. Ying, and Y. Fan-ding, "Incremental Mining Alogorithm Pre-FP in Association Rules Based on FP-tree," in *First International Conference on Networking and Distributed Computing*, 2010, pp. 199–203.

[5] J. Pieprzyk and B. Sadeghiyan, *Design of Hashing Algorithms*. Berlin: Springer Berlin Heidelberg, 2006.

[6] V. Kashyp and A. K. Gupta, "A Review on Modern Approach: New Parameter for Recent Improvement of Apriori Algorithm," *International Journal of Computer Science and Mobile Computing*, vol. 4, no. 12, 2015.

[7] C. C. Aggarwal, *Data Mining: The Textbook*. Switzerland: Springer, 2015.

[8] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed. San Fransisco: Elsevier Science & Technology, 2006.

[9] D. T. Larose, *Data Mining Method and Model*. Canada: Wiley, 2007.

[10] V. Kotu and B. Deshpande, *Predictive Analytics and Data Mining: Concepts and Practice with RapidMiner*. Burlington: Morgan Kaufmann, 2014.

[11] R. Rathinasabapathy and R. Bhaskaran, "Performance Comparison of Hashing Algorithm with Apriori," in *International Conference on Advances in Computing, Control, and Telecommunication Technologies*, 2009, pp. 729–733.

[12] J. S. Park, M.-S. Chen, and P. S. Yu, "An effective hash-based algorithm for mining association rules," *ACM SIGMOD Record*, vol. 24, no. 2, pp. 175–186, May 1995.

[13] A. Bhandari, A. Gupta, and D. Das, "Improvised Apriori Algorithm Using Frequent Pattern Tree for Real Time Applications in Data Mining," *Procedia Computer Science*, vol. 46, pp. 644–651, 2015.

[14] B. M. Rao and S. Aguru, "A Hash Based Frequent Itemset Mining using Rehashing," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 2, no. 12, 2014.

[15] A. M. J. M. Z. Rahman, P. Balasubramanie, and P. V. Krihsna, "A Hash based Mining Algorithm for Maximal Frequent Item Sets using Linear Probing," *Journal of Computer Science*, vol. 8, no. 1, pp. 14–19, 2009.